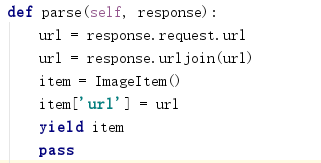
1.构造请求，定义图片URL的最大爬取数 MAX\_IMAGES，也就是Excel表里数据的行数，将其定义在settings.py里。引入 openpyxl库，读取Excel表里的数据加入到request中，由于Excel表中图片的URL有的没有 ‘https:’前缀，有的有，所以用re正则匹配URL中的 ‘https:’，将其替换为‘ ’，再在前面加上‘http:’，统一格式。 用for i in range(2, self.settings.get('MAX\_IMAGES') + 2):和 for j in range(1, 2):两个循环只需修改range中的参数就可以读取到任意行数间和任意列间的数据。

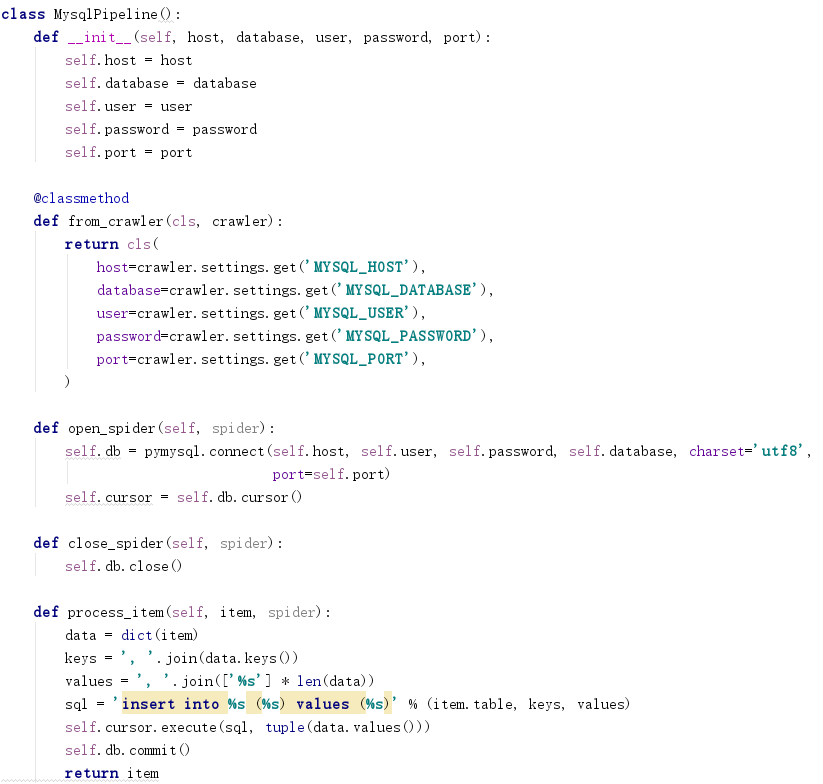
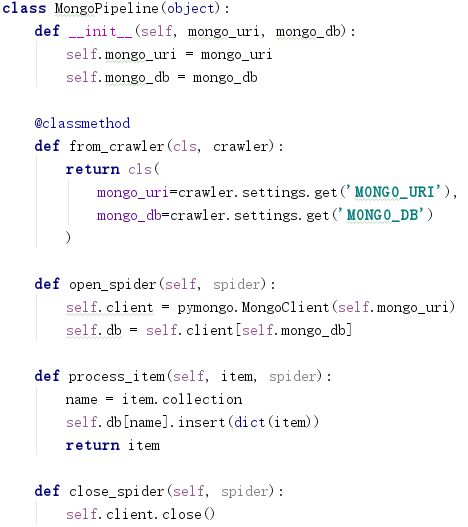


2.在settings.py中修改 ROBOTSTXT\_OBEY的值为False，表示不遵守爬虫协议，即 robots 协议，用来限定爬虫程序可以爬取的内容范围。通常写在 robots.txt 文件中，该文件保存在网站的服务器上，爬虫程序访问网站时首先查看此文件。为了使爬虫可以正常爬取，故作此修改。

3.提取信息，定义一个imageitem， 两个属性collection = table = 'images'，分别表示MongoDB中collection的名称和MySQL中的表名，两个字段url = Field()，flag = Field()，表示下载的图片的URL和它的状态标志，ok表示图片下载成功，false表示下载失败。用parse()方法提取spider里的有关信息



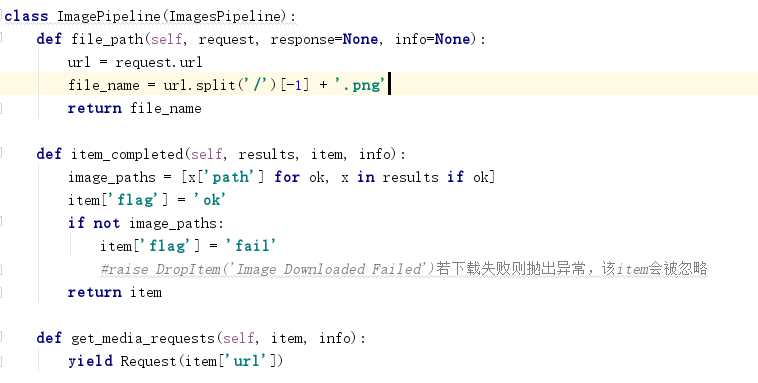
4.存储信息，在 pipelines.py中，添加数据库存储的类，相关的数据库连接参数都定义在settings.py



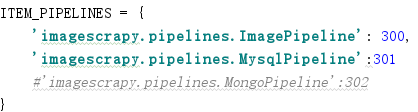
5.实现imagepipeline， scrapy有专门用于下载的pipeline，包括文件和图片，支持异步与多线程。首先定义一个下载图片保存到本地的路径，在settings.py中添加 IMAGES\_STORE = './images'，即保存到当前路径下的images文件夹里。

file\_path函数用来返回保存的文件名，用split函数分隔URL并提取最后一部分作为文件名。 item\_completed函数是单个item完成下载时的处理方法，result返回是该item的下载结果，为一个列表，列表中每一元素皆为一个元组，包含了下载的信息，遍历下载结果找出所有成功的下载列表。若列表为空，则表示对应item的图片下载失败，将其flag标志位改为false，在数据库通过查询flag标志位可快速查看哪些图片下载成功或失败。

get\_media\_requests函数其中item是爬取生成的item对象，因为scrapy是先爬取URL再进行下载的，所以直接将爬取到的URL字段提取出来生成request对象，加入到调度队列，等待被调度，进行下载。

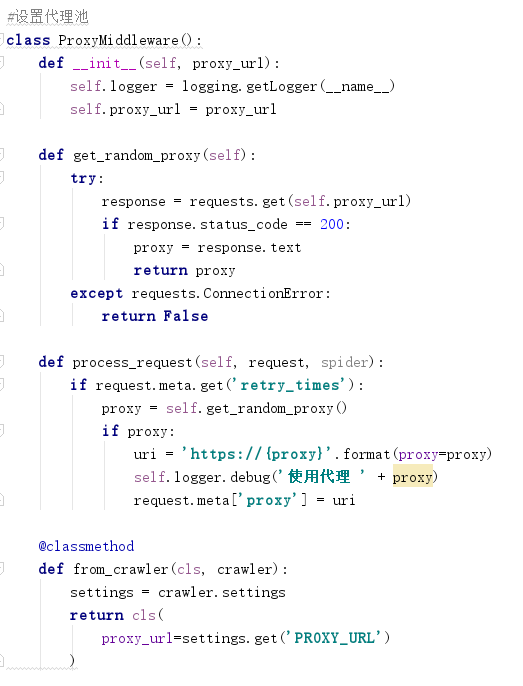


6.启用pipeline，在settings.py设置 ITEM\_PIPELINES，调度顺序按优先值越小越高来排，首先调用imagepipeline处理下载图片的item，再将下载信息存到数据库



7.反爬处理，在 middlewares.py中设置代理池和user-agent池，在settings.py中启用，随机从池中抽取代理和 user-agent，防止爬虫被禁





8.scrapy速度优化，在settings.py中进行相关配置。

增加并发：默认scrapy开启的并发线程的个数是32个，可以适当的进行增加，设置CONCURRENT\_REQUESTS=100

禁止cookie：如果不是真的需要cookie的话，可以在scrapy爬取的时候禁止cookie从而来减少CPU的使用率，提升爬取的效率，COOKIES\_ENABLED=False

